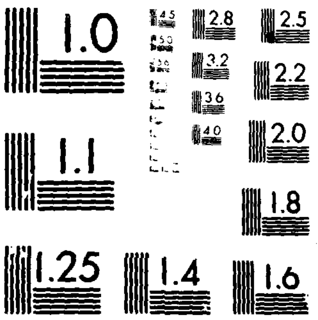
AD-A081 991   GEORGIA INST OF TECH ATLANTA COMPUTER ARCHITECTURE LAB   F/G 9/2
MACROSTRUCTURE LOGIC ARRAYS.(U)
OCT 79 C O ALFORD, M R MCQUADE   DAS660-78-C-0137
UNCLASSIFIED   NL

1 OF 1
AD
A081991

END
DATE
FILMED
4-80
DTIC


MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

6 MACROSTRUCTURE LOGIC ARRAYS

2

CAL GTEE

9 Final ~~Report~~ Submitted to:

Ballistic Missile Defense Advanced Technology Center

Contract No. 15 DASG60-78-C-0137 new

11 8 October ~~8~~, 1979

COMPUTER ARCHITECTURE LABORATORY
SCHOOL OF ELECTRICAL ENGINEERING
GEORGIA INSTITUTE OF TECHNOLOGY
ATLANTA, GEORGIA 30332

12 45

Mr. B. Kelley
BMDATC
Project Engineer

10 Cecil O. Alford
Professor of
Electrical Engineering

M. R. McQuade
Ph.D. Student
School of Electrical Engineering

411 656

DISCLAIMER

Accession For	
NTIS GRA&I	✓
DDC TAB	☐
Unannounced	☐
Justification Ltr on file	
By	
Distribution/	
Availability Codes	
Dist	Avail and/or special
A	

I. INTRODUCTION

The objective of this research was to investigate the concept, design and performance analysis of a computing structure to solve linear ordinary differential equations. Previous efforts in this area have been classified as Digital Differential Analyzers. This research is not an attempt to extend this prior effort, but rather to develop an entirely new approach [1-14]. This report summarizes the results of the research in four parts; (1) the concept for a digital computing structure, (2) the design of the computing structure, (3) the performance of the computing structure and, (4) conclusions and recommendations for further research.

II. CONCEPT

A. General

Many physical systems can be modeled by linear ordinary differential equations of order N. This equation can be solved using an analog computer or a general purpose digital computer. The analog computer has a speed advantage since it solves the problem using parallel computing elements. The digital computer yields a more accurate solution, and eliminates any scaling problems by using floating point arithmetic. However, the digital computer is slower than the analog computer, and in most cases, is more difficult for the user to access [15]. The research goal was to design and build a digital computing structure which encompasses the advantages of both these computers (except for floating point arithmetic) and avoids the difficulties of each.

B. Mathematical Forms and Solutions

A third order linear ordinary differential equation can be expressed as

$$\dddot{y}(t) + a_2\ddot{y}(t) + a_1\dot{y}(t) + a_0y(t) = b_2\ddot{X}(t) + b_1\dot{X}(t) + b_0X(t)$$

$$\ddot{y}(o) = C_2, \quad \dot{y}(o) = C_1, \quad y(o) = C_0 \qquad (1)$$

If any of the coefficients a_i or b_i are functions of t, this equation is time varying. The following does not treat this case, but can be extended to do so. In order to solve (1) using an analog computer, it is necessary to program the computer to connect the hardware as shown in Figure 1. This structure will then yield y(t) as a function of the input X(t). It is important to note that the hardware elements operate in parallel resulting in high computation speed for the analog computer. It should also be noted that the third order differential equation in (1) and the complementary solution structure in Figure 1 can be extended to a higher order system. Since this does not add to this discussion, the extension is not carried out in this report.

A common transformation to convert (1) to a set of first order differential equations is [16]

$$\begin{bmatrix} \dot{Z}_1(t) \\ \dot{Z}_2(t) \\ \dot{Z}_3(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -a_0 & -a_1 & -a_2 \end{bmatrix} \begin{bmatrix} Z_1(t) \\ Z_2(t) \\ Z_3(t) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} X(t)$$

$$Y(t) = \begin{bmatrix} b_2 & b_1 & b_0 \end{bmatrix} \begin{bmatrix} Z_1(t) \\ Z_2(t) \\ Z_3(t) \end{bmatrix} + \begin{bmatrix} 0 \end{bmatrix} X(t) \qquad (2)$$

This set of differential equations can be solved using the structure shown in Figure 2. In general, the integrators would be digital processors and the multiplier and adder blocks would also be digital hardware. This structure in Figure 2 is rather straightforward for a third order differential equation. However, as the order increases, the bottom row of a_i coefficients in (2) will also increase. This implies a larger collection of multipliers and adders for the implementation in Figure 2, which also increases the solution time.

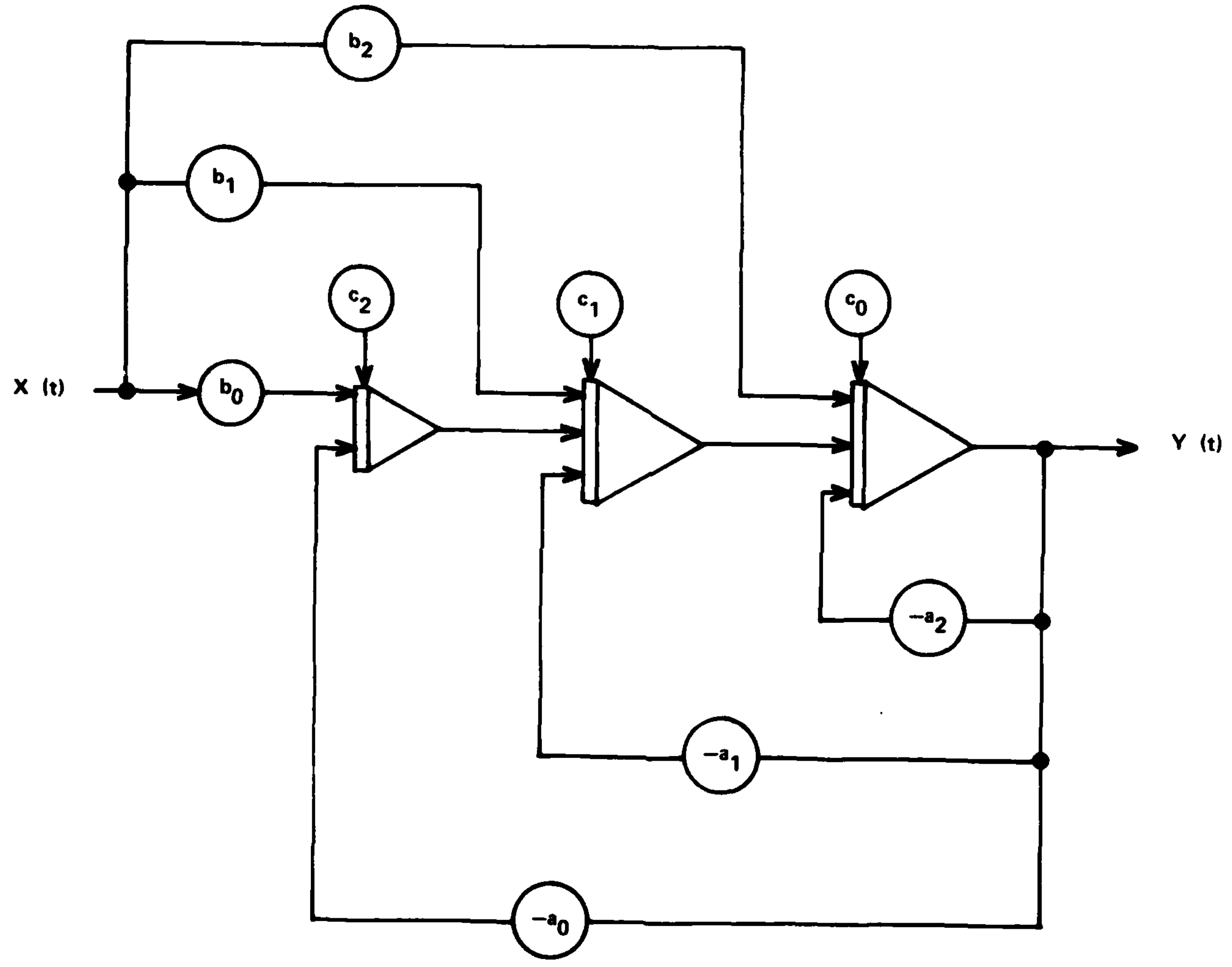


Figure 1. ANALOG COMPUTER SOLUTION OF EQUATION (1)

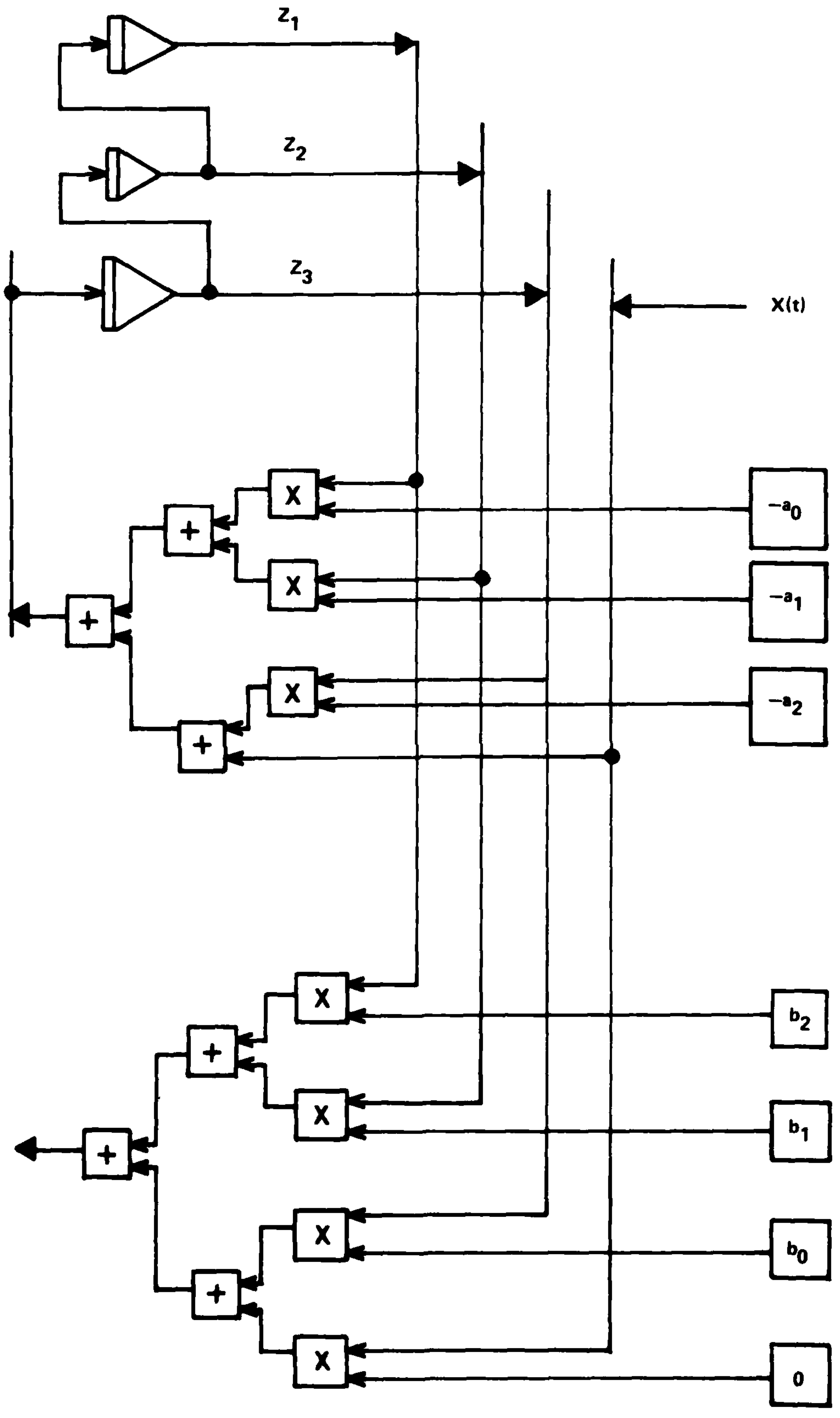


Figure 2. Parallel Digital Solution of Equation (2)

An alternate transformation on (1) will yield

$$\begin{bmatrix} \dot{Z}_1(t) \\ \dot{Z}_2(t) \\ \dot{Z}_3(t) \end{bmatrix} = \begin{bmatrix} -a_2 & 1 & 0 \\ -a_1 & 0 & 1 \\ -a_0 & 0 & 0 \end{bmatrix} \begin{bmatrix} Z_1(t) \\ Z_2(t) \\ Z_3(t) \end{bmatrix} + \begin{bmatrix} b_2 \\ b_1 \\ b_0 \end{bmatrix} X(t)$$

$$Y(t) = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} Z_1(t) \\ Z_2(t) \\ Z_3(t) \end{bmatrix} + \begin{bmatrix} 0 \end{bmatrix} X(t) \qquad (3)$$

with a solution structure as shown in Figure 3. This is the form selected for SPOCK since the multiplier-adder group can be built with a set of modules where each module has two multipliers and two adders. This results in a higher performance machine and some savings in hardware. The design of the SPOCK structure, based on this mathematical formulation of the differential equation, will be given in the next section.

III. DESIGN OF THE COMPUTING STRUCTURE

A. Architecture

1. Organization: The structure for SPOCK I is shown in Figure 4. There are five modules; namely, (1) the processor, (2) input bus interface, (3) output bus interface, (4) function, and (5) coefficient register modules. In addition, a host computer is used to initialize the units, load the necessary routines, and monitor the results. A bus controller is shown but has not been implemented in hardware. This device is primarily a sequencer and controls the gating between all units and the busses.

2. Processor Module: Each processor is a 16-bit, fixed point computer constructed from the Intel 3000 integrated circuit family [17]. Each processor

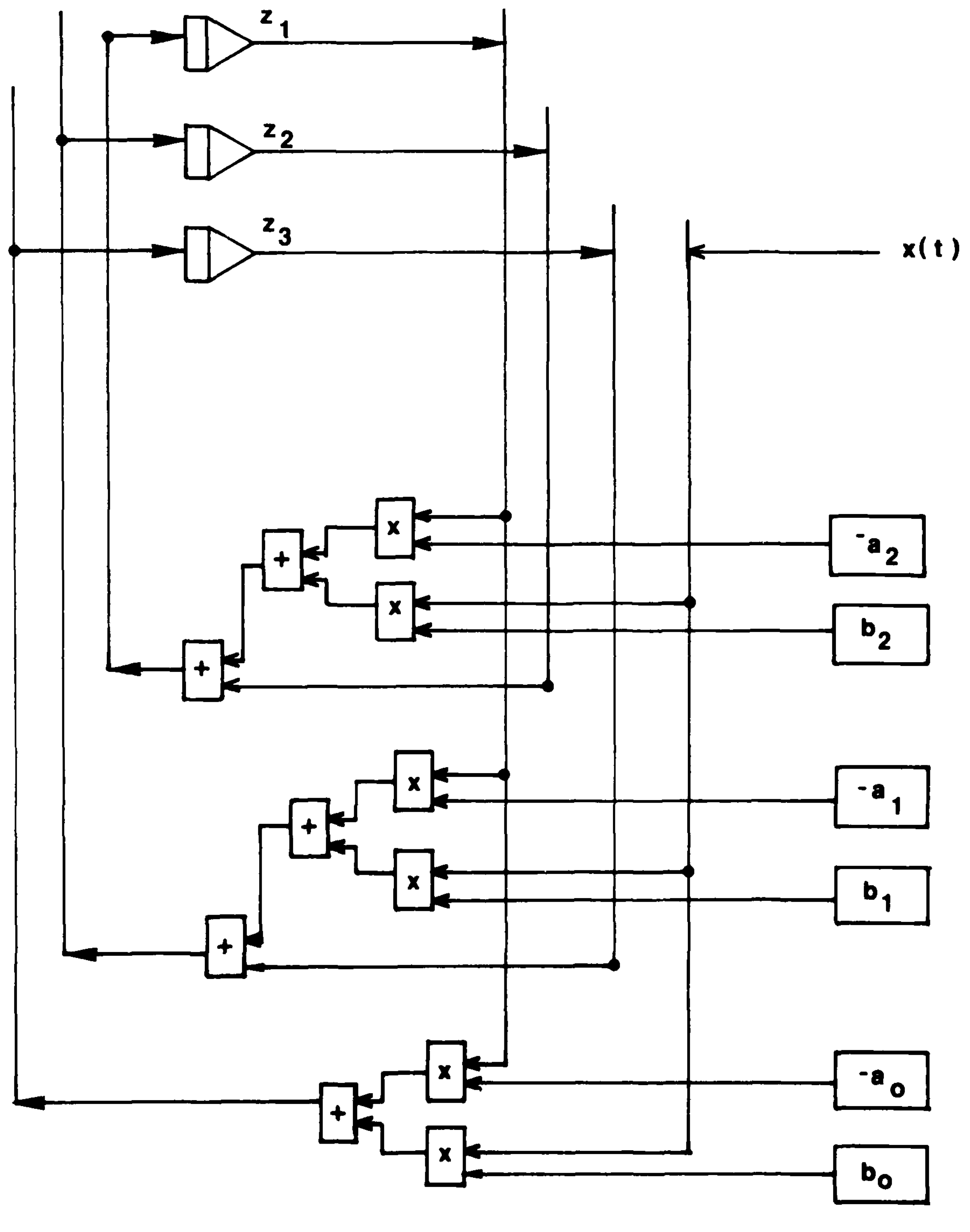


FIGURE 3 PARALLEL DIGITAL SOLUTION OF EQUATION (3)

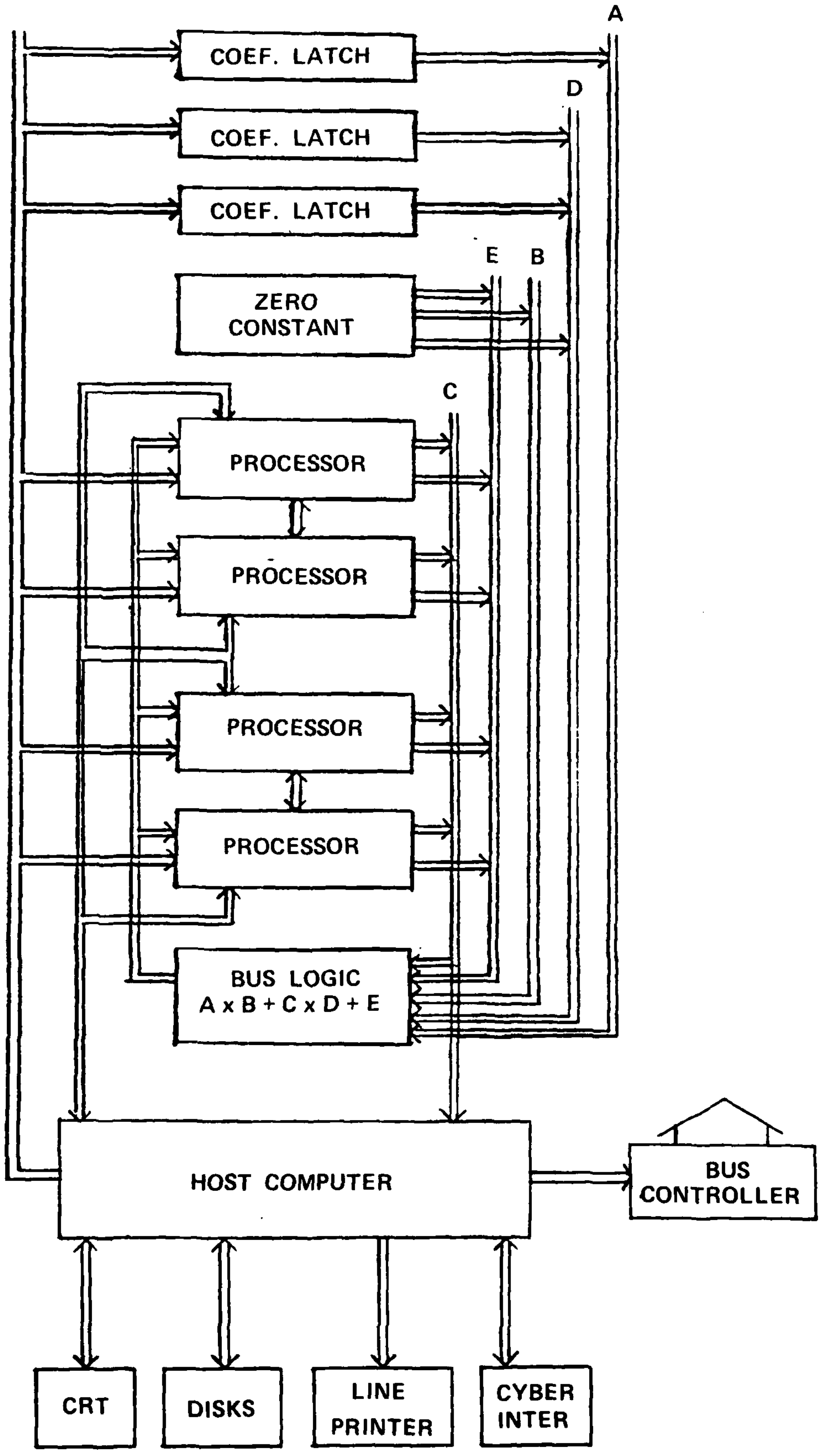


FIGURE 4 SPOCK I COMPUTING STRUCTURE

has an auxiliary multiplier (16 x 16) to enhance performance and a 1024 word RAM for microinstruction storage. A RAM was used instead of a ROM to enable the loading of various function routines. Thus, each processor can be configured as any specific function by a proper choice of microinstructions. All tests were carried out by selecting the function to be an integrator and using various numerical schemes to approximate the desired integration function. The processor module is shown in Figure 5. For full parallel operation, one processor module is needed for each state variable.

3. Function Module: The function module performs the calculation

$$F = AxB + CxD + E \tag{3}$$

where F is the output and A, B, C, D and E are digital values on five input busses. Parallel operation requires one function module for each state variable. The structure for the function module is shown in Figure 6.

4. Bus Interface Modules: The output bus interface module is used to latch each processor output. This value can then be applied to any one of N busses by selection of the latch control. The input interface module is used to latch the outputs from the function units. One of these is then selected as the input to each processor using the latch control lines. The number of modules required is equal to the number of processors. The bus interface modules are shown in Figures 7 and 8.

5. Coefficient Register Module: The constant or coefficient registers are used to store the a_i and b_i constants in the state equation matrix. These modules are register latches in the SPOCK structure. They are loaded with the appropriate constant when the problem is initialized. The output of each latch is applied to the appropriate bus using a sequencer to control the latch enable lines. The module structure is shown in Figure 9.

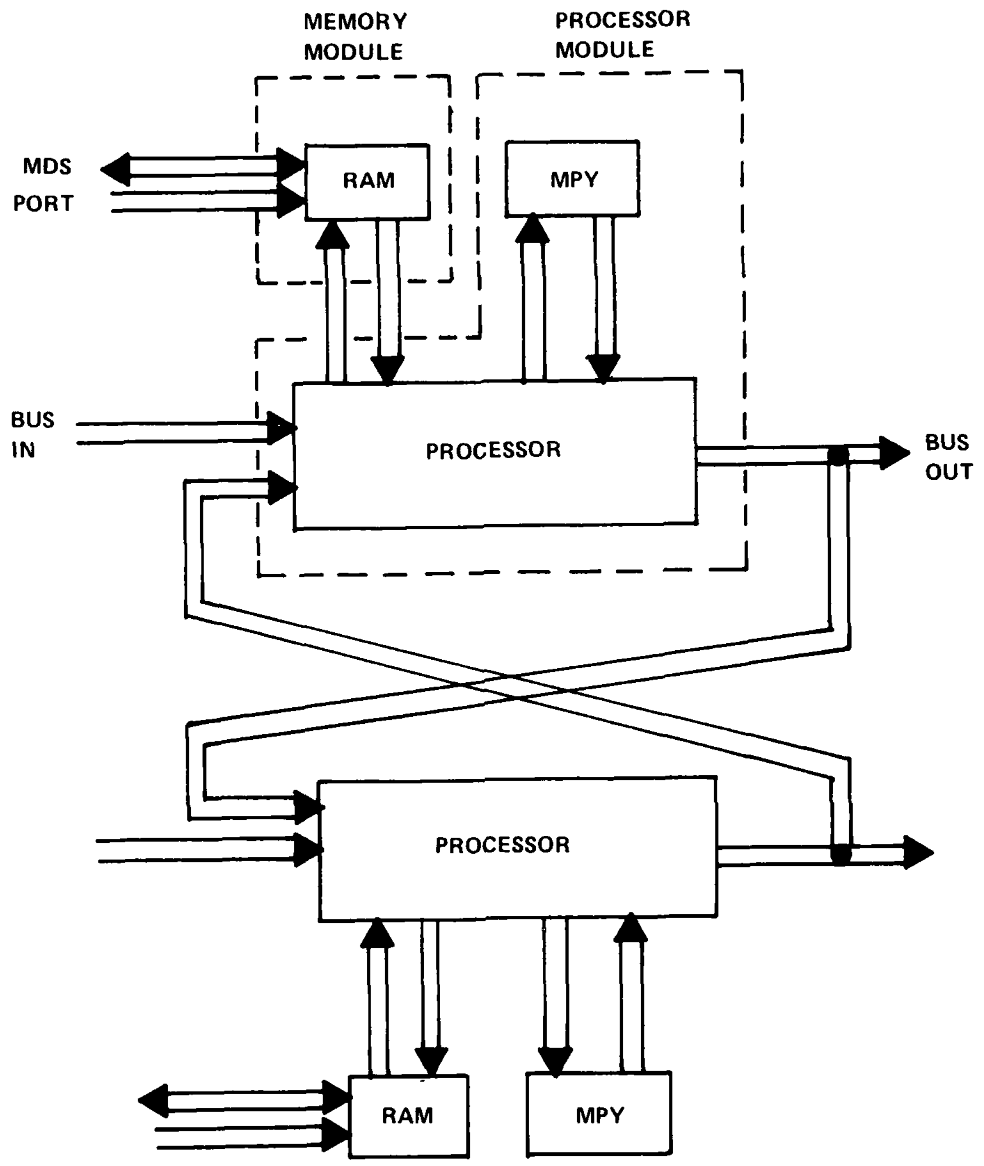


Figure 5. SPOCK Processor/Memory Module

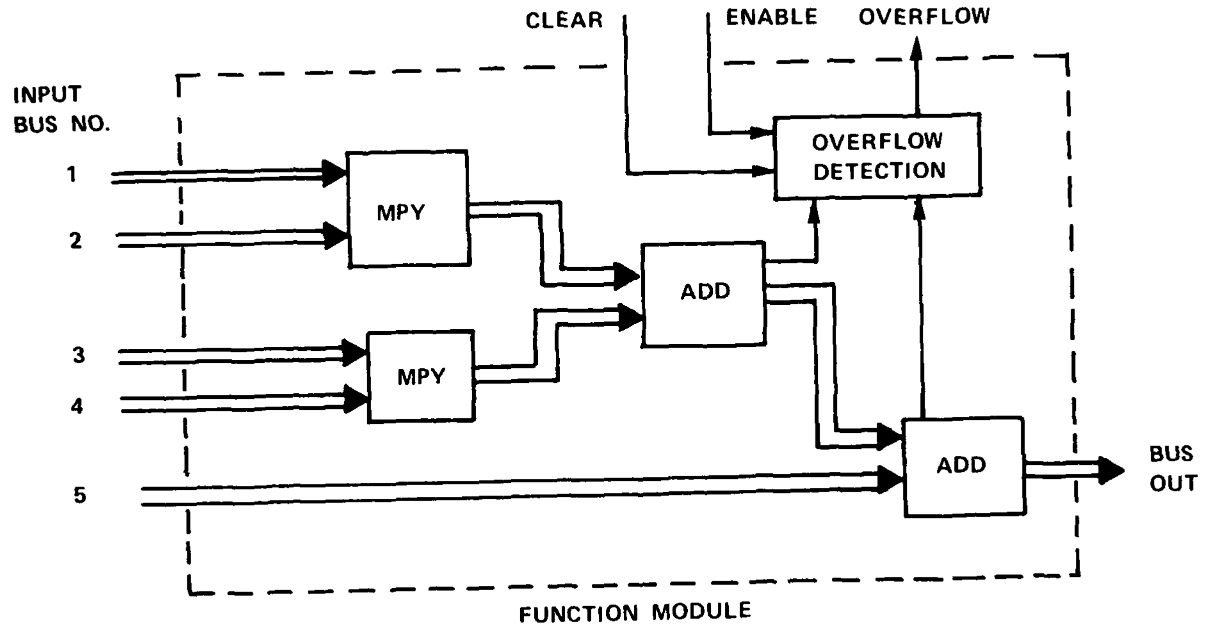


Figure 6. SPOCK Function Module

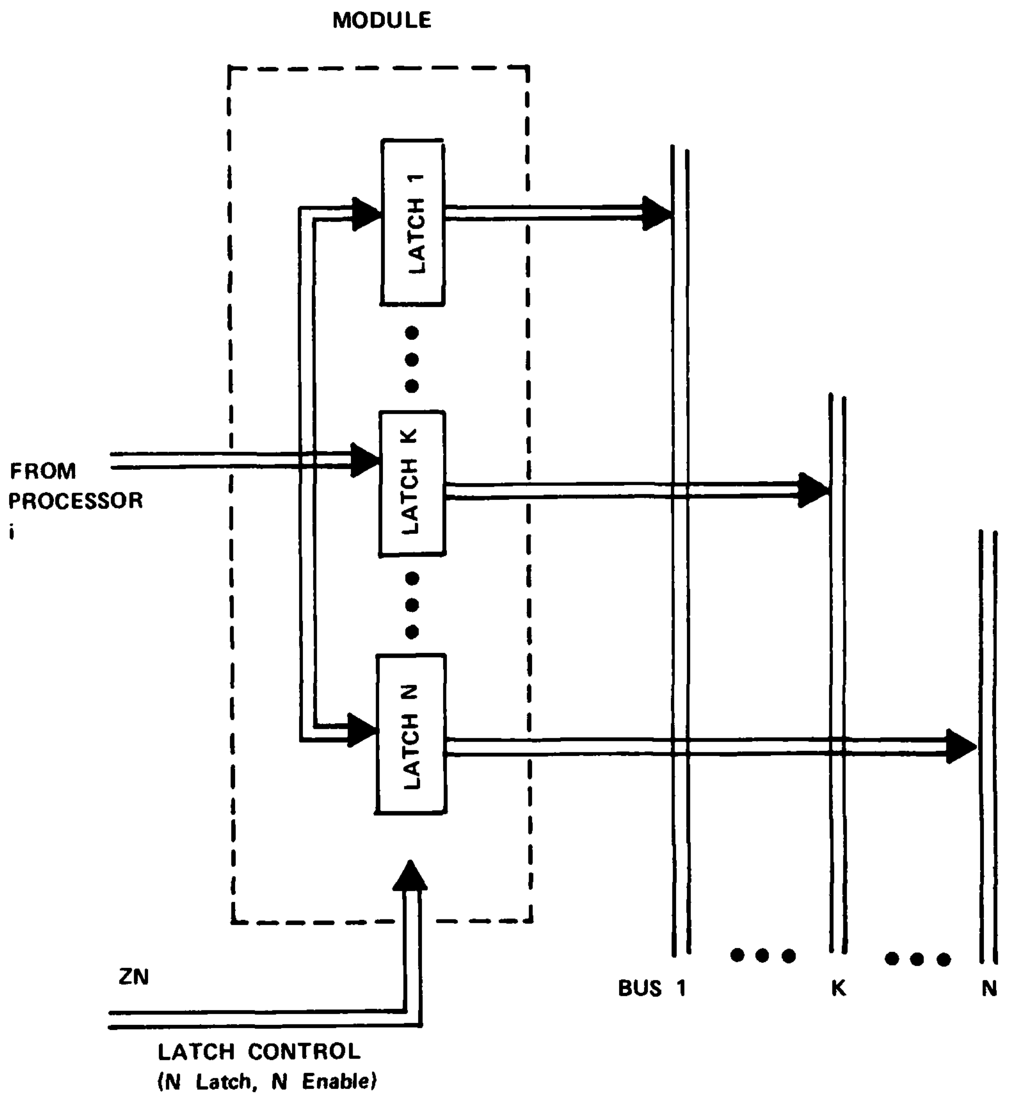


Figure 7. Output Bus Interface Module

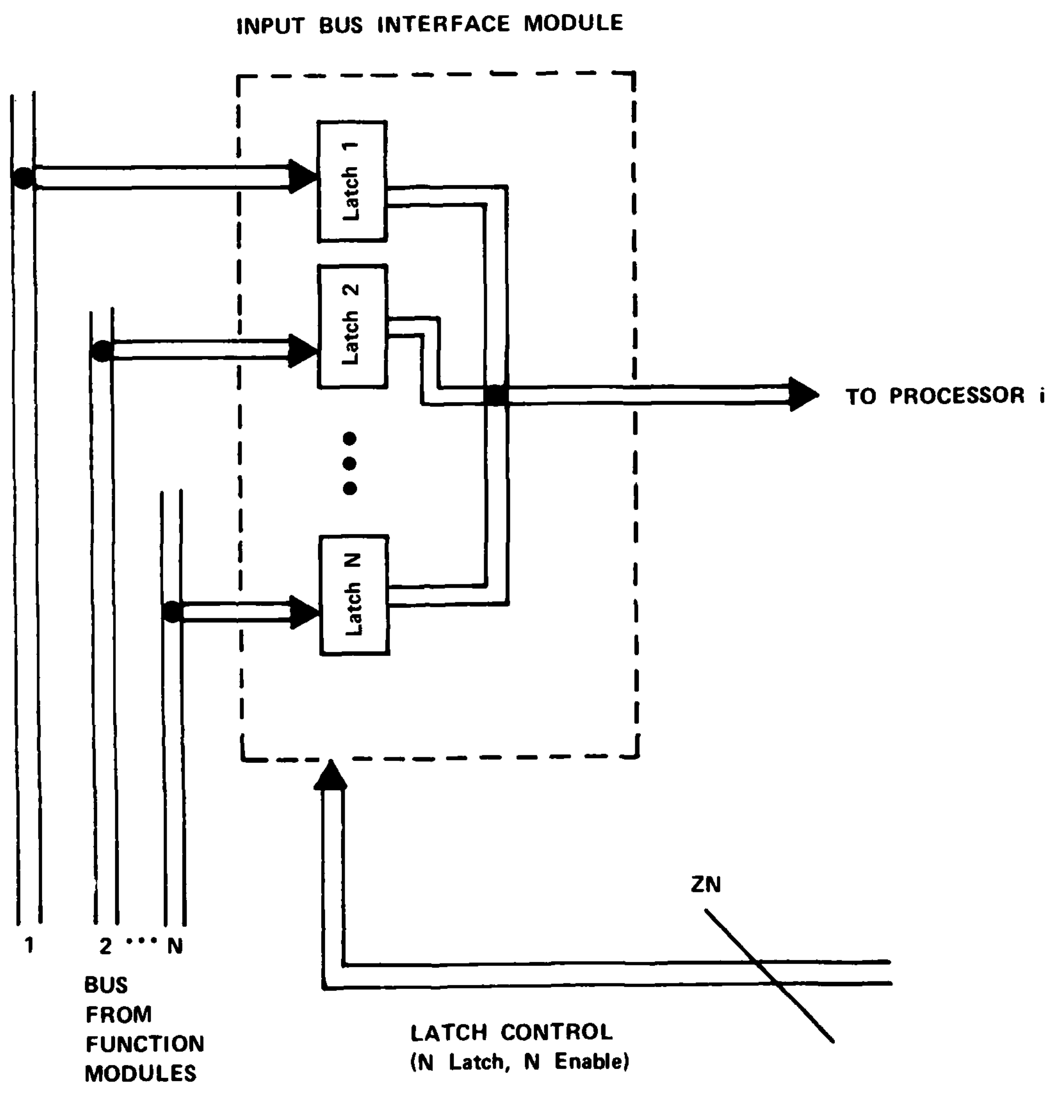


Figure 8. Input Bus Interface Module

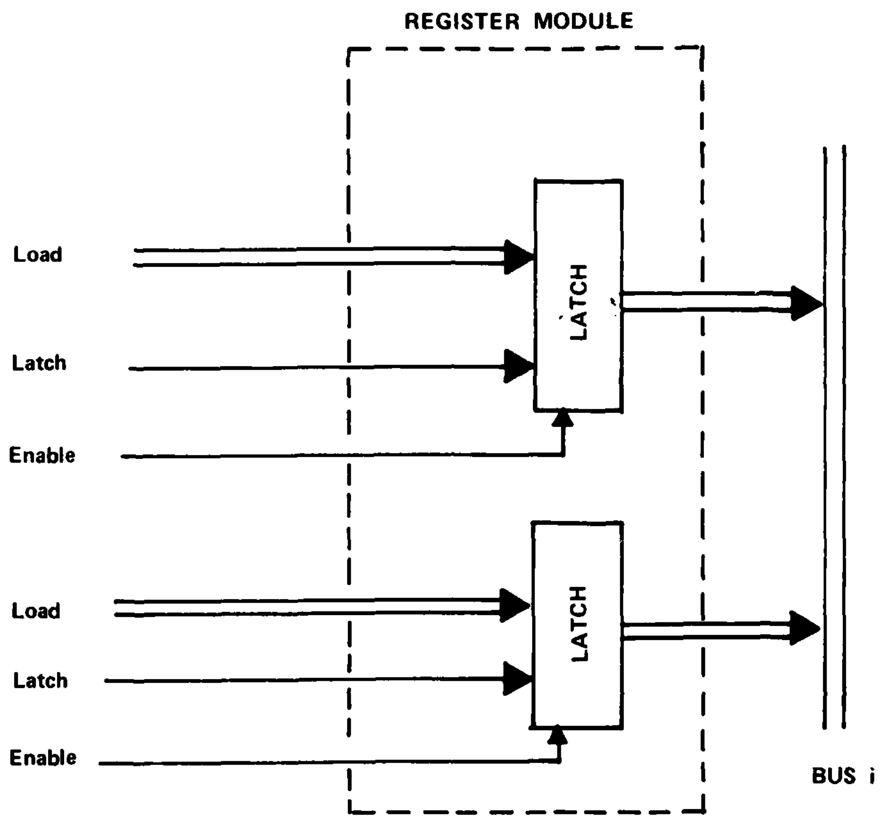


Figure 9. Coefficient Register Module

If equation (1) is permitted to have time varying coefficients, it will be necessary to change the constant registers to a circular queue. The values would then be shifted and gated in much the same way the current configuration handles the constants. The change would require loading of all the discrete points when the problem is initialized, and control information to perform the shifting operation.

IV. PERFORMANCE ANALYSIS

A. Microprogram

Each of the nine integration schemes given in Table 1 has been coded into a microprogram for the SPOCK processors. These integration routines have complexities which are reflected in the program size. This program size is given in Table 2 as the number of microinstructions required to implement the routine. For example, twelve microinstructions are required to implement the Modified-Euler routine. Thus, each integration step requires the execution of these twelve instructions.

The various integration routines execute their microinstructions sequentially except for the Parallel Adams-Bashforth Predictor/Corrector. This routine is implemented such that the predictor (5 microinstructions) can be computed in parallel with the corrector (10 microinstructions) using two processors. This is the only "parallel" integration routine which was implemented. Performance results will reflect this parallel versus sequential implementation.

B. Function Evaluations

The evaluation of an integration step requires the evaluation of the function,

$$\dot{Y}_n = f\,(Y_n, X_n) \qquad (4)$$

to obtain the new value Y_{n+1}. From Table 1 it can be seen that the new point, Y_{n+1}, can be evaluated for any of the first seven routines if prior points have been saved and $\dot{Y}_n$ is available. This function evaluation requires the use of

Table 1. Numerical Integration Methods

Euler

$$Y_{n+1} = Y_n + T\dot{Y}_n$$

Modified Euler

$$Y^p_{n+1} = Y^c_n + T\dot{Y}^c_n$$

$$Y^c_{n+1} = Y^c_n + \frac{T}{2}(\dot{Y}^p_{n+1} + \dot{Y}^c_n)$$

Adams-Bashforth Two Point Predictor

$$Y_{n+1} = Y_n + T\dot{Y}_n + \frac{T}{2}(\dot{Y}_n - \dot{Y}_{n-1})$$

A - B Two Point & Trapezoidal Corrector

$$Y^p_{n+1} = Y^c_n + T\dot{Y}^c_n + \frac{T}{2}(\dot{Y}^c_n - \dot{Y}^c_{n-1})$$

$$Y^c_{n+1} = Y^c_n + \frac{T}{2}(\dot{Y}^p_{n+1} + \dot{Y}^c_n)$$

Parallel Predictor Corrector

$$Y^p_{n+1} = Y^c_{n-1} + 2T\dot{Y}^p_n$$

$$Y^c_n = Y^c_{n-1} + \frac{T}{2}(\dot{Y}^p_n + \dot{Y}^c_{n-1})$$

Adam's Three Point Predictor

$$Y_{n+1} = Y_n + \frac{23T}{12}\dot{Y}_n - \frac{4T}{3}\dot{Y}_{n-1} + \frac{5T}{12}\dot{Y}_{n-2}$$

Adam's Pair Three Point

$$Y^p_{n+1} = Y^c_n + \frac{23T}{12}\dot{Y}^c_n - \frac{4T}{3}\dot{Y}^c_{n-1} + \frac{5T}{12}\dot{Y}^c_{n-2}$$

$$Y^c_{n+1} = Y^c_n + \frac{5T}{12}\dot{Y}^p_{n+1} + \frac{2T}{3}\dot{Y}^c_n - \frac{T}{12}\dot{Y}^c_{n-1}$$

Table 1. Numerical Integration Methods (Cont'd)

Three Point Runge-Kutta

$$Y_{n+1} = Y_n + T(\frac{1}{4}\dot{Y}_n + 0Y_{n+\frac{1}{3}} + \frac{3}{4}\dot{Y}_{n+\frac{2}{3}})$$

$$\dot{Y}_n = F(X_n, Y_n)$$

$$\dot{Y}_n + \frac{1}{3} = F(X_n + \frac{T}{3}, Y_n + \frac{T}{3}\dot{Y}_n$$

$$\dot{Y}_{n+\frac{2}{3}} = F(X_n + \frac{2T}{3}, Y_n + \frac{2T}{3}\dot{Y}_{n+\frac{1}{3}}$$

Four Point Runge-Kutta

$$Y_{n+1} = Y_n + T(\frac{1}{6}\dot{Y}_n + \frac{1}{3}\dot{Y}_{n+\frac{1}{2}_1} + \frac{1}{3}\dot{Y}_{n+\frac{1}{2}_2} + \frac{1}{6}\dot{Y}_{n+1})$$

$$\dot{Y}_n = F(X_n, Y_n)$$

$$\dot{Y}_{n+\frac{1}{2}_1} = F(X_n + \frac{T}{2}, Y_n + \frac{T}{2}\dot{Y}_n)$$

$$\dot{Y}_{n+\frac{1}{2}_2} = F(X_n + \frac{T}{2}, Y_n + \frac{T}{2}\dot{Y}_{n+\frac{1}{2}_1})$$

$$\dot{Y}_{n+1} = F(X_n + T, Y_n + T\dot{Y}_{n+\frac{1}{2}_2}$$

Table 2. Program Characteristics for Numerical Methods

Method	Microprogram Size	No. of Function Evaluations
Euler	4	1
Modified Euler	12	2
2nd A-B Pred.	8	1
2nd A-B P/C	22	2

Parallel P/C	5(P)
	10(C)
		1
		1

3rd Adam's Pred.	14	1
3rd Adam's P/C	27	2
3rd Runge-Kutta	18	3
4th Runge-Kutta	27	4

the function network to solve one row of the SPOCK state equation. For Euler's method, one pass through the function network is required since $\dot{Y}_n$ can be used with a stored value of Y_n to compute Y_{n+1}. However, for the Modified-Euler method, the function network must be used twice; once to compute $\dot{Y}^c_n$ which is used to compute Y^p_{n+1}, and $\dot{Y}^p_{n+1}$ which can be used to generate Y^c_{n+1}. The Parallel Adams-Bashforth P/C is again a special case in that the two function evaluations can be done in parallel by using two hardware units.

The Runge-Kutta methods differ from the other methods in that function values cannot be stored to reduce the computation at each step. These methods require a number of intermediate values in going from point nT to point (n+1)T. Each of these values represents a function evaluation resulting in a total of three for the third order method and four for the fourth order method.

The required number of function evaluations per integration step is given in Table 2 for each of the nine methods. These function evaluations impact performance due to the computation delay. For most methods, this is a sequential delay which is added to the integration time. For the Modified-Euler, the delay can be broken down into the components shown in Figure 10. Assume the routine has just computed Y^c_n. The next four steps will be: (1) function computation for $\dot{y}^c_n$, (2) integration step for y^p_{n+1}, (3) function computation for $\dot{y}^p_{n+1}$, and (4) integration step for y^c_{n+1}. Each of these is shown in Figure 10.

C. Performance Equation

1. Definitions: A performance equation can be derived based on the delays discussed in the two previous sections. The performance is related to the solution of a state vector of N first order differential equations. It is first necessary to define the following parameters:

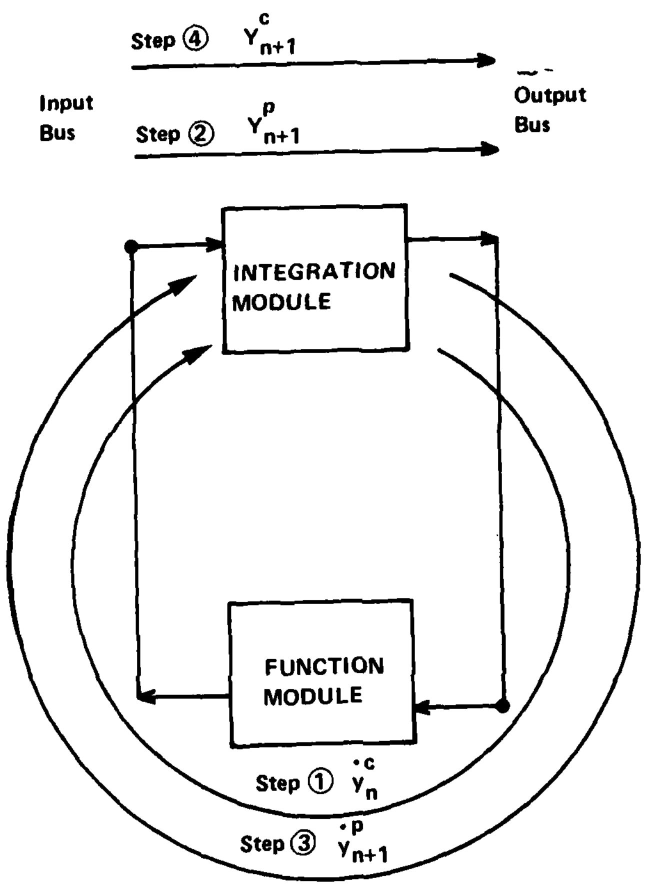


Figure 10. Computation Delays for Modified Euler Integration Step

S_T	-	Time to compute one integration step for the state vector
D_F	-	Function Unit Delay
D_I	-	Processor Delay in Computing one Integral Step
K	-	Number of passes through the function unit for one integration step
O_I	-	Processor Overlap with function unit
N	-	Number of state equations
I	-	Number of processor modules
F	-	Number of function unit modules.

The processor delay, D_I, is a function of the number of microinstructions and the time it takes to execute each instruction. For predictor-corrector methods, this delay could be separated into the predictor delay, D_{IP}, and the corrector delay, D_{IC}. However, when these two are added the result is D_I.

2. Overlap Time: The overlap time, O_I, represents the time associated with certain integration routines which can be hidden by doing the calculation in parallel with the function unit. For example, the Adams Three Point Predictor is given by

$$Y_{n+1} = Y_n + \frac{23T}{12} \dot{Y}_n - \frac{4T}{3} \dot{Y}_{n-1} + \frac{5T}{12} \dot{Y}_{n-2} \tag{6}$$

While the function unit computes $\dot{Y}_n$, the processor can begin the computation based on the terms which are already known. In this case, the overlap time would be equal to D_F since several multiplies and adds can be performed before $\dot{Y}_n$ is needed. It should be noted that

$$O_I \leq D_F \qquad \text{and} \qquad O_I \leq D_I \tag{7}$$

3. Performance Equations: To illustrate the development of the performance equations three separate cases will be considered using the Modified-Euler integration technique. In the first case N = I = F. For this case, Figure 10

indicates the sequence for one integration step. This sequence and the respective delays are

Step 1	$\dot{y}^c_n$	D_F
Step 2	y^p_{n+1}	D_{IP}
Step 3	$\dot{y}^p_{n+1}$	D_F
Step 4	y^c_{n+1}	D_{IC}

The resulting time is then

$$S_T = 2\ D_F + D_{IP} + D_{IC} \tag{8}$$

Extending to the general solution and including the possibility of overlap with the function units yields

$$S_T = KD_F + (D_I - O_I) \tag{9}$$

For this particular integration routine, O_I would be zero.

In case two, let N = 3, I = 2, F = 3. The computation sequence and the delays are then

Step 1	$\dot{y}^c_{1,n}$	$\dot{y}^c_{2,n}$	$\dot{y}^c_{3,n}$	D_F
Step 2	$y^p_{1,n+1}$	$y^p_{2,n+1}$		D_{IP}
Step 3	$y^p_{3,n+1}$			D_{IP}
Step 4	$\dot{y}^p_{1,\ n+1}$	$\dot{y}^p_{2,\ n+1}$	$\dot{y}^p_{3,\ n+1}$	D_F
Step 5	$y^c_{1,n+1}$	$y^c_{2,n+1}$		D_{IC}
Step 6	$y^c_{3,\ n+1}$			D_{IC}

The computation time for the state vector is

$$S_{TI} = 2D_F + 2D_{IP} + 2D_{IC} \tag{10}$$

where S_{TI} is the solution time for the multiplexed processor case.

In general, this equation becomes

$$S_{TI} = KD_F + D_I \left\lceil \frac{N}{I} \right\rceil \tag{11}$$

where $\lceil X \rceil$ represents the smallest integer $\geq X$.

For the third case, let N = 3, I = 3, F = 2. The computation sequence and the associated delays are

Step 1	$\dot{y}^c_{1,n}$	$\dot{y}^c_{2,n}$		D_F
Step 2	$\dot{y}^c_{3,n}$			D_F
Step 3	$y^p_{1,n+1}$	$y^p_{2,n+1}$	$y^p_{3,n+1}$	D_{IP}
Step 4	$\dot{y}^p_{1,n+1}$	$\dot{y}^p_{2,n+1}$		D_F
Step 5	$\dot{y}^p_{3,n+1}$			D_F
Step 6	$y^c_{1,n+1}$	$y^c_{2,n+1}$	$y^c_{3,n+1}$	D_{IC}

The computation time for the state vector is

$$\begin{aligned} S_{TF} &= 4D_F + D_{IP} + D_{IC} \\ &= 4D_F + D_I \end{aligned} \tag{12}$$

where S_{TF} is the solution time for the multiplexed function unit case. In general, this equation becomes

$$S_{TF} = K \left\lceil \frac{N}{F} \right\rceil D_F + D_I \tag{13}$$

Equations (9), (11), and (13) represent the three cases for any of the integration routines. In (11) and (13), the term O_I should be subtracted as in (9). When this is included, the multiplexed times can be normalized to give

$$\frac{S_{TI}}{S_T} = \frac{1 + P \left\lceil \frac{N}{I} \right\rceil}{1+P} \tag{14}$$

and

$$\frac{S_{TF}}{S_T} = \frac{\left\lceil \frac{N}{F} \right\rceil + P}{1+P} \tag{15}$$

where

$$P = \frac{D_I - O_I}{KD_F} \tag{16}$$

Equations (14), (15), and (16) indicate the relative time penalty when solving N equations with insufficient integrators or function units. Another case which includes an insufficient number of both units can be obtained in a similar manner as

$$S_{TIF} = D_I \left\lceil \frac{N}{I} \right\rceil + KD_F \left\lceil \frac{N}{F} \right\rceil \tag{17}$$

Forming a relative ratio gives

$$S_{TIF} = \frac{(D_I - O_I) \left\lceil \frac{N}{F} \right\rceil + KD_F \left\lceil \frac{N}{F} \right\rceil}{KD_F + D_I - O_I} \tag{18}$$

$$= \frac{P \left\lceil \frac{N}{I} \right\rceil + \left\lceil \frac{N}{F} \right\rceil}{1 + P} \tag{19}$$

The important point is that the fully parallel solution time is given by equation (19) for one increment in the independent variable. When F = I = 1 the structure reduces to a sequential structure and the relative time becomes

$$\frac{S_{TIF}}{ST} = \frac{P \left\lceil \frac{N}{1} \right\rceil + \left\lceil \frac{N}{1} \right\rceil}{1 + P} \tag{20}$$

$$= N \tag{21}$$

When N = I = F, the relative time becomes

$$\frac{S_{TIF}}{ST} = \frac{P \left\lceil \frac{N}{N} \right\rceil + \left\lceil \frac{N}{N} \right\rceil}{1 + P} \tag{22}$$

$$= 1 \tag{23}$$

Other forms of multiplexing the hardware will yield relative time increases of

$$S_T \leq S_{TIF} \leq N\ S_T \tag{24}$$

D. Error Analysis

1. Truncation Errors: When using a numerical integration routine, a truncation error is introduced. This error is related to the difference between the given numerical method and the Taylor Series for the expansion of a function [18]. Typically, numerical methods will agree with the Taylor Series through the first few terms. The first term which doesn't agree can be used to establish a bound on the computation error. These bounds are shown in Table 3 for the first seven numerical methods. Runge-Kutta truncation errors are more difficult to express in this compact form. An alternate error technique is used for these methods, called Richardson's Extrapolation [19].

To develop these error expressions, let y_n and y_{n+1} be the true solutions at steps n and n+1. In order to go from step n to n+1, a single step, h, can be used once or a step of h/2 can be used twice. The solution when the step is used twice is y_2 and the solution when the step is used once is y_1. These solutions generate errors given by

$$y_{n+1} - y_1 \simeq 2^{r+1}\ C_n\ h^{r+1} \simeq E_1 \tag{25}$$

$$y_{n+1} - y_2 \simeq 2\ \ C_n\ h^{r+1} \simeq E_2 \tag{26}$$

where r is the order of the numerical method. When C_n is eliminated, the result is

$$y_{n+1} - y = E_2 = \frac{y_2 - y_1}{2^r - 1} \tag{27}$$

This equation can be used for any of the numerical methods, but is particularly attractive for Runge-Kutta algorithms.

Table 3. Truncation Errors for Numerical Methods

Method	Truncation Error $nT < \varepsilon < (n+1)T$
Euler	$\frac{T^2}{2} \ddot{Y} (\varepsilon)$
Modified Euler	$-\frac{T^3}{12} \dddot{Y} (\varepsilon)$
2nd Adams Bashforth Predictor	$\frac{5T^3}{12} \dddot{Y} (\varepsilon)$
2nd Adams Bashforth Predictor/Corrector	$-\frac{T^3}{12} \dddot{Y} (\varepsilon)$
Parallel Predictor/ Corrector	$\frac{T^3}{3} \dddot{Y} (\varepsilon)$
3rd Adams Predictor	$\frac{9T^4}{24} \ddddot{Y} (\varepsilon)$
3rd Adams Predictor/ Corrector	$\frac{-T^4}{24} \ddddot{Y} (\varepsilon)$

The errors for third and fourth order Runge-Kutta methods are then

$$ERR_{RK3} \leq \frac{y_2 - y_1}{7} \tag{28}$$

$$ERR_{RK4} \leq \frac{y_2 - y_1}{15} \tag{29}$$

where ERR_{RK3} and ERR_{RK4} represent the local truncation error in the respective Runge-Kutta methods.

2. Round-Off Errors: Another source of error is due to finite precision arithmetic. This error is called round-off [20]. In the SPOCK computer, all data and calculations are carried out in 16-bit, twos complement arithmetic. One of the objectives of the research was to establish the relative importance of round-off error compared to truncation error.

3. Error Curves: In order to get some performance results, a particular equation was selected. The equation

$$\begin{aligned} &\ddot{y}(t) + 2\,(0.139999)\,\dot{y}(t) + y(t) = 0 \\ &y(0) = 0, \quad \dot{y}(0) = 0.75 \end{aligned} \tag{30}$$

was solved on SPOCK using all nine integration methods. True values to (30) were obtained using the exact solution and 120-bit floating point arithmetic. The difference between these two solutions gives the error generated by SPOCK when solving (30). This set of error curves is shown in Figure 11 as a function of step size. The error which is plotted is

$$ERR_{TOT} = \Delta T \sum_{i=1}^{n} |Y120_i - YSPOCK_i|^2 \tag{31}$$

where ΔT is the step size, $Y120_i$ represents the true solution at step i, $YSPOCK_i$ is the SPOCK solution at step i, and ERR_{TOT} is the total error. The step size was selected by setting $\Delta T = 2\pi / 12 \cdot 2^i$ where $i = 0, 1, 2, \ldots, 7$.

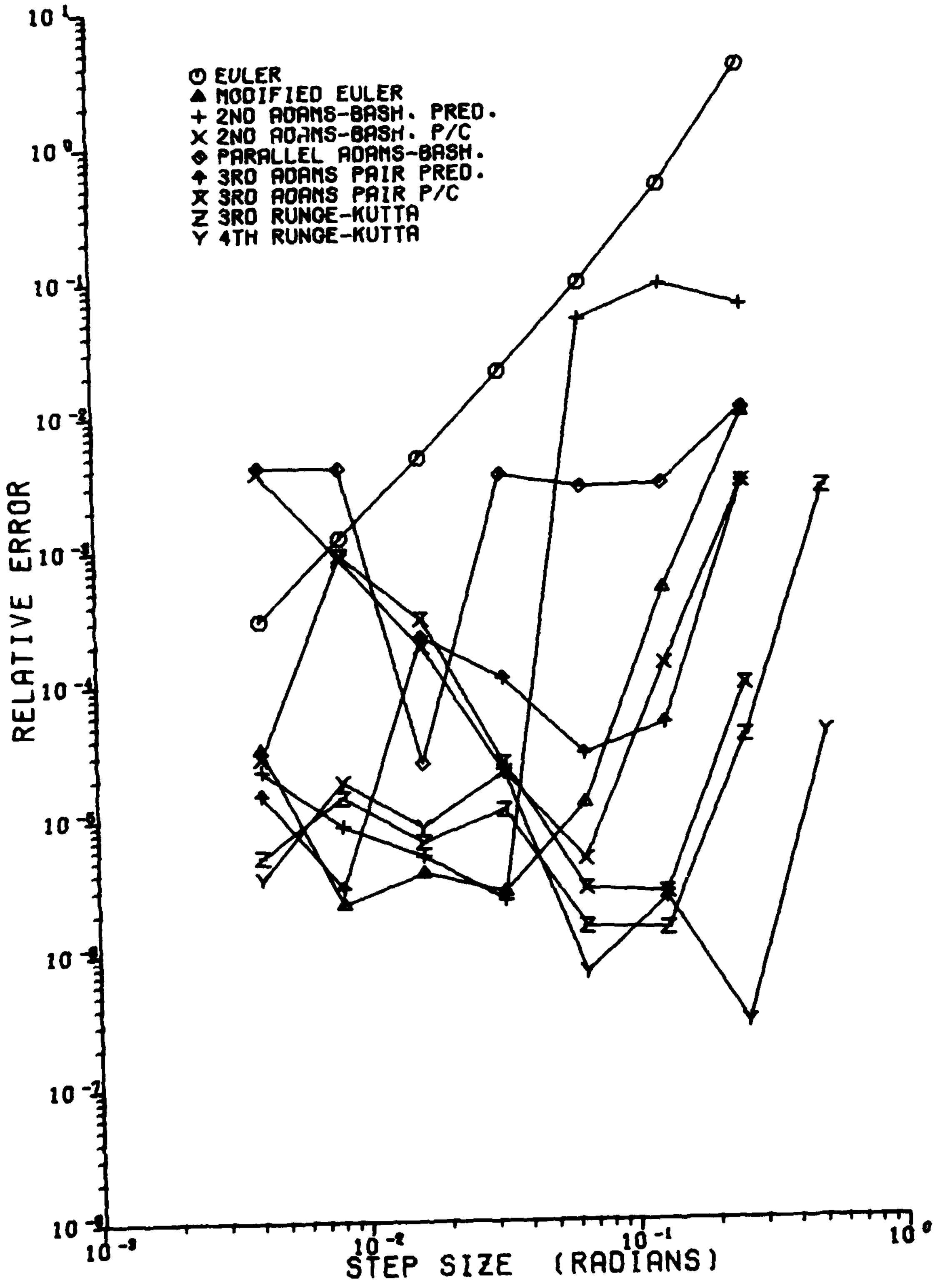


Figure 11. SPOCK I SOLUTION ERROR

In order to separate the error into its two components, equation (30) was solved by each of the numerical methods using 120 bit floating point arithmetic. The exact values were also calculated using 120 bits. Roundoff error in this case was assumed to be negligible. The truncation error was calculated by

$$ERR_{TOT} = \Delta T \sum_{i=1}^{n} |Y120_i - YNUM_i|^2 \tag{32}$$

where ΔT is the step size.

$Y120_i$ is the true value at step i, $YNUM_i$ is the value at step i using a particular numerical method and ERR_{TRU} is the truncation error. The set of error curves produced by (32) is shown in Figure 12.

The round-off error can now be determined by forming:

$$ERR_{RO} = ERR_{TOT} - ERR_{TRU} \tag{33}$$

where ERR_{RO} is the round-off error for a particular numerical method using the SPOCK hardware. These error curves are shown in Figure 13. The superiority of the higher order methods can be made more specific by the following performance analysis.

E. Relative Performance

In order to assess the capability of SPOCK, the following hardware parameters are used:

$O_I = 0$;

D_I = (Number Microinstructions) x 250 nsec.

D_F = 500 nsec.

Using the time parameters given above, the various methods have a solution time per step as indicated in Table 4. For example, the Modified-Euler method has a

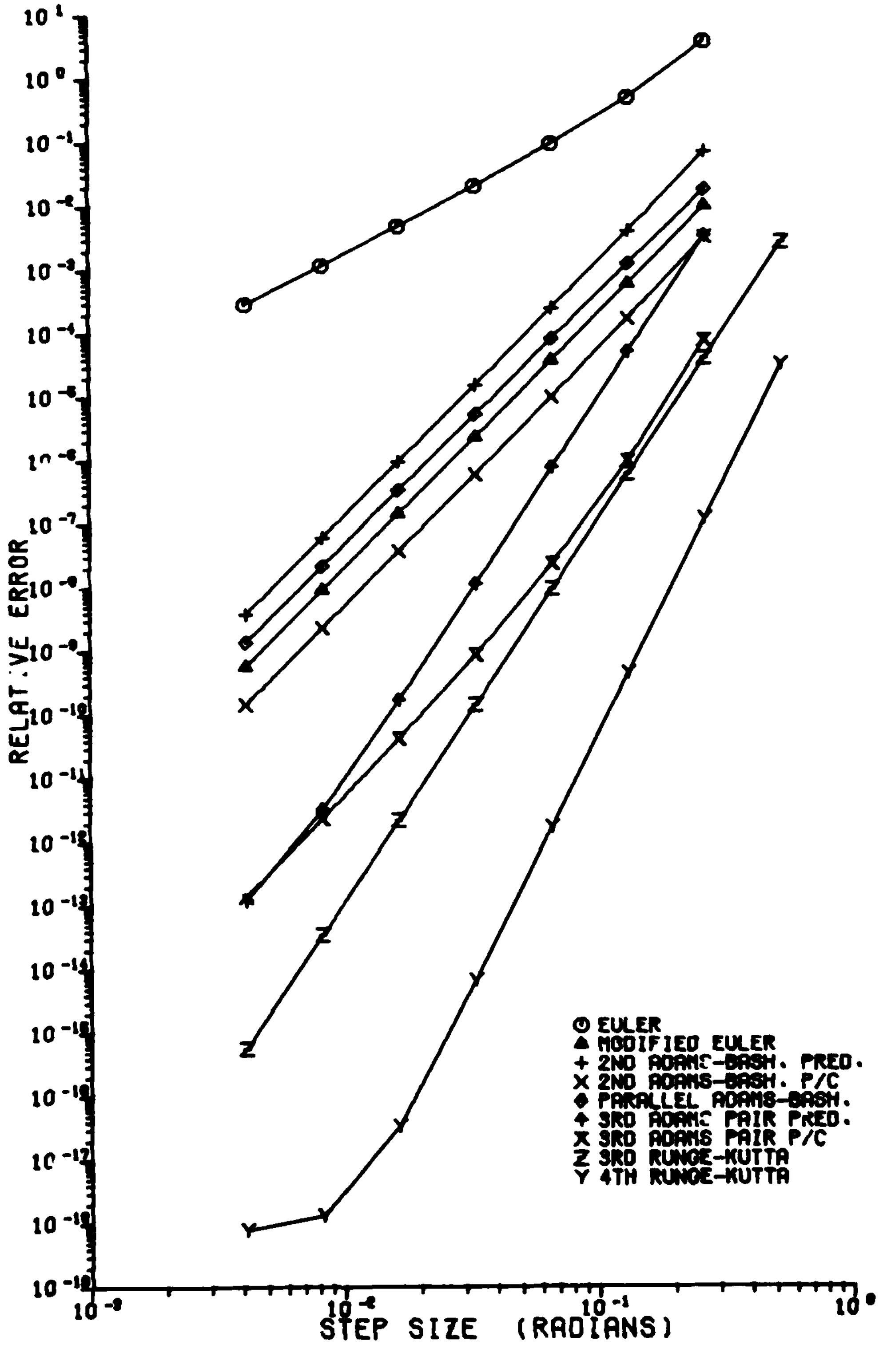


Figure 12. ACCUMULATED TRUNCATION ERROR

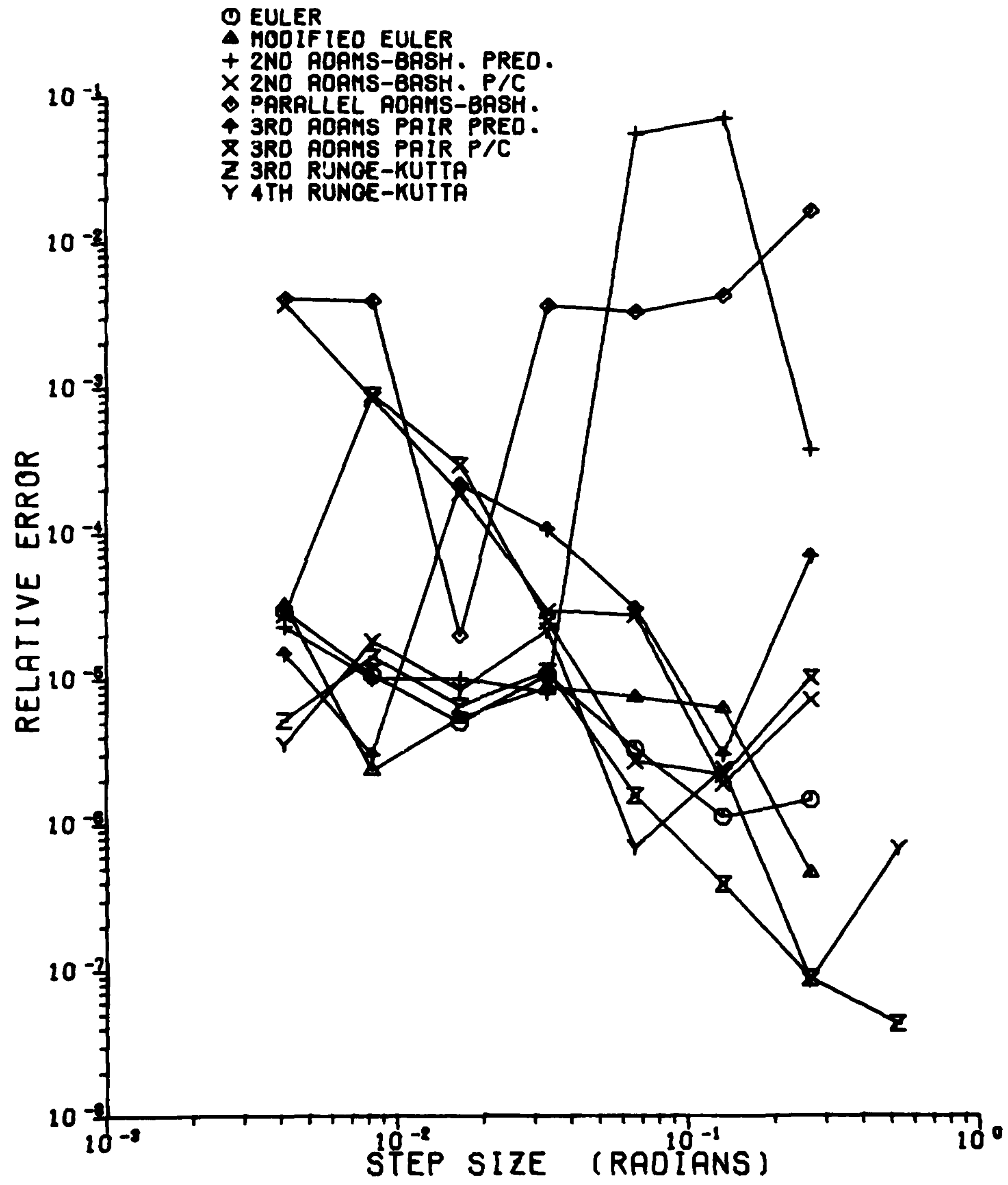


Figure 13. ACCUMULATED ROUND-OFF ERROR

solution time per step of:

$$S_T = KD_F + D_I - O_I$$

$$= (2)\ (500) + (12)\ (250) \qquad (34)$$

$$= 4000 \text{ nsec.}$$

The error curves can be used to pick the maximum step size for each numerical method. In order to set the step size, the absolute error as a percent of the maximum value (1.0 for the SPOCK machine) was used. A typical error curve is shown in Figure 14 for the Modified-Euler numerical method with T = 0.1309. This shows the maximum error to be approximately 0.6%. When T = 0.262 for this numerical method, the maximum error exceeds 1%. Thus, for the performance analysis, the maximum step size for the Modified-Euler method is set at T = 0.1309. A similar analysis for the other routines leads to the maximum step sizes given in Table 4.

The maximum step size for the Modified-Euler method is $T_{max} = 0.1309$ radians. Thus, the maximum frequency using the Modified-Euler method is

$$f_{max} = \frac{0.1309 \times 10^6}{2\pi \times 4.0} = 5208 \text{ hz.} \qquad (35)$$

A similar calculation can be carried out for each method to give the upper frequency limit shown in Table 4. It should be emphasized that this limit also includes the requirement that the maximum absolute error will be less than 1% over the solution range.

F. Stability and Convergence

The error in a numerical method is related to the stability and convergence of the method. It is generally reported in the literature that stable methods give good solutions [21, 22]. A more accurate statement is that a stable method gives a solution which converges to the true solution for large values of the independent variable.

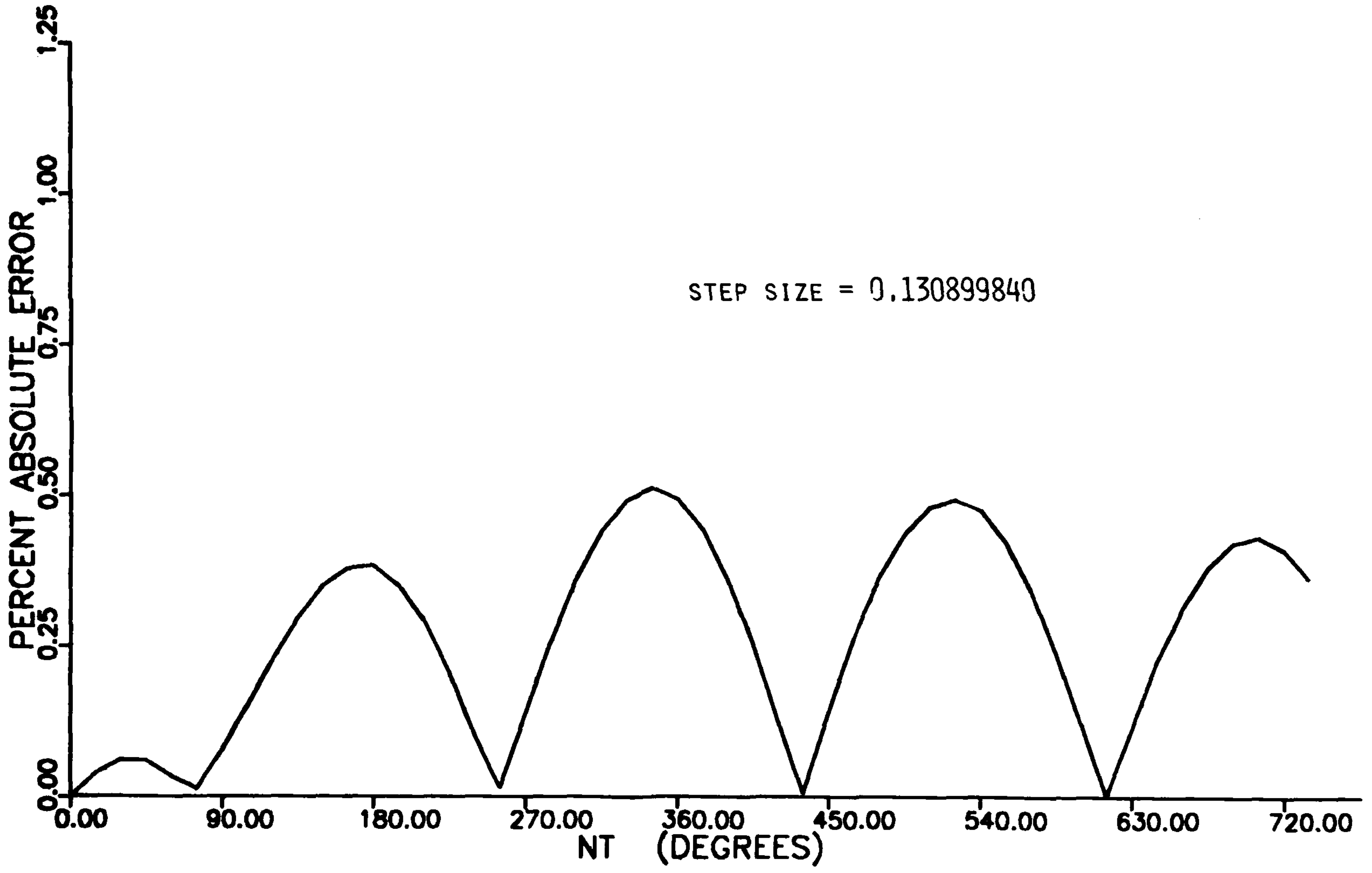


Figure 14. SPOCK I ABSOLUTE ERROR USING MODIFIED-EULER INTEGRATION METHOD

Table 4. SPOCK Performance Analysis

Method	Time per Step (μsec.)	T_{max} (Radians)	Maximum Frequency (Hz)
Euler	1.5	0.0041	434
Modified Euler	4.0	0.131	5208
2nd A-B Predictor	2.5	0.0327	2083
2nd A-B Predictor/ Corrector	6.5	0.131	3205
Parallel P/C	3.0	0.0163	868
3rd Adams Predictor	4.0	0.131	5208
3rd Adams Predictor/ Corrector	7.75	0.262	5376
3rd Runge-Kutta	6.0	0.262	6944
4th Runge-Kutta	8.75	0.524	9524

For example, assume it is desired to solve the equation

$$\dot{y}(t) + y(t) = 10u(t), \; Y(0) = 0 \tag{36}$$

using Euler's method. Substituting for $\dot{y}(t)$ gives

$$\frac{y_{n+1} - y_n}{T} + y_n = 10u(n), \; Y(0) = 0 \tag{37}$$

where T is the integration step size. Transforming to the Z-domain yields

$$zY(z) = -T\, y(z) + y(z) + \frac{10Tz}{z-1} \tag{38}$$

$$y(z) \;\; [\, z + T - 1 \,] = \frac{10Tz}{z-1} \tag{39}$$

$$y(z) = \frac{10Tz}{(z-1)\;(z+T-1)} \tag{40}$$

$$= \frac{10Tz}{(z-1)\;(z - (1-T))} \tag{41}$$

The solution for Y(z) is stable provided the roots of the denominator lie within the unit circle (the point Z = 1 is a special case). Thus, for stability

$$0 \leq T < 2. \tag{42}$$

The solution to (41) is

$$y^*(nT) = 10 \;\; [1 - (1-T)^n] \tag{43}$$

The true solution to (36) is

$$y(t) = 10\;(1-e^{-t}), \; t \geq 0 \tag{44}$$

If this solution is examined at discrete points, the values are

$$y(nT) = 10\;(1-e^{-nT}) \;, \; n \geq 0 \tag{44}$$

The error at discrete points is

$$\text{ERROR}_n = |y(nT) - y^*(nT)| \;\; = \;\; |10\;(1-T)^n - 10e^{-nT}| \tag{46}$$

Note that lim T →0 for the error is zero at all finite values of nT. But also note that when T ≠ 0, errors exist at all points t = nT. An important concept is the relative error [23] defined as

$$|\text{REL ERROR}| = \frac{|\text{ERROR}_n|}{|y(nT)|} = \frac{|(1-T)^n - e^{-nT}|}{|1-e^{-nT}|} \qquad (47)$$

$$|\text{REL ERROR}| = \left|\frac{(1-T)^n - e^{-nT}}{1-e^{-nT}}\right| \qquad (48)$$

For distinct values of T the following relative errors exist

$$T = 0.1 \qquad |\text{REL ERROR}| = \frac{|(.9)^n - e^{-.1n}|}{|1 - e^{-.1n}|} < 1 \; ; \; n > 0$$

$$T = 0.5 \qquad |\text{REL ERROR}| = \frac{|(.5)^n - e^{-.5n}|}{|1 - e^{-.5n}|} < 1 \; ; \; n > 0$$

$$T = 1.0 \qquad |\text{REL ERROR}| = \frac{|-e^{-n}|}{|1-e^{-n}|} < 1 \; ; \; n > 0$$

$$T = 1.5 \qquad |\text{REL ERROR}| = \frac{|(.5)^n - e^{-1.5n}|}{|1-e^{-1.5n}|} < 1 \; ; \; n > 0$$

$$T = 2.0 \qquad |\text{REL ERROR}| = \frac{|(-1)^n - e^{-2n}|}{|1-e^{-2n}|} \geq 1 \; ; \; n > 0$$

These values indicate for values of T except T = 2 the relative error is less than one. This implies the approximate solution is "close" to the true solution or converges to the true solution. It is evident, however, that for T = 0.5 an error exists for all finite values of n. As $n \to \infty$ in this case, the relative error $\to 0$. A similar situation exists for T = 0.1, T = 1.0, T = 1.5, and in general all T which yield a stable solution. Hence, stability implies a solution that converges to the true solution as $t \to \infty$. It does not imply an accurate solution for finite values of $t = nT$.

Accuracy requires an examination of the relative error. For example, consider the two cases T = 0.1 and T = 1.0, both of which give stable solutions.

The relative errors for these two cases at the same time points are

$$|\text{REL ERROR } (T = 0.1)| = \left| \frac{(.9)^{10n} - e^{-n}}{1-e^{-n}} \right| \quad ; \quad n = 1, 2, \ldots$$

$$|\text{REL ERROR } (T = 1.0)| = \left| \frac{-e^{-n}}{1-e^{-n}} \right| \quad ; \quad n = 1, 2, \ldots$$

which shows

$$|\text{REL ERROR } (T = 0.1)| < |\text{REL ERROR } (T = 1.0)|$$

Hence, the choice of $T = 0.1$ produces a more accurate solution at each point $t = nT$ than the choice of $T = 1.0$.

In general, stability of the solution is not the driving requirement on the choice of step size. Accuracy at each step will usually dictate a much smaller step size than the stability requirement. If the solutions to (36) using $T = 0.5$, 1.0 and 1.5 are compared to the true solution as in Figure 15, it is obvious that certain solutions are very inaccurate. These solutions are stable and will converge to the exact solution for large values of nT. However, in simulation work, this is seldom the case of interest. The performance analysis section used a more conventional accuracy requirement based on the error as a percent of the maximum value. This is the typical performance specification used by analog/hybrid computers [24]. Further, the error limit of 1% of the maximum value is also consistent with analog/hybrid hardware.

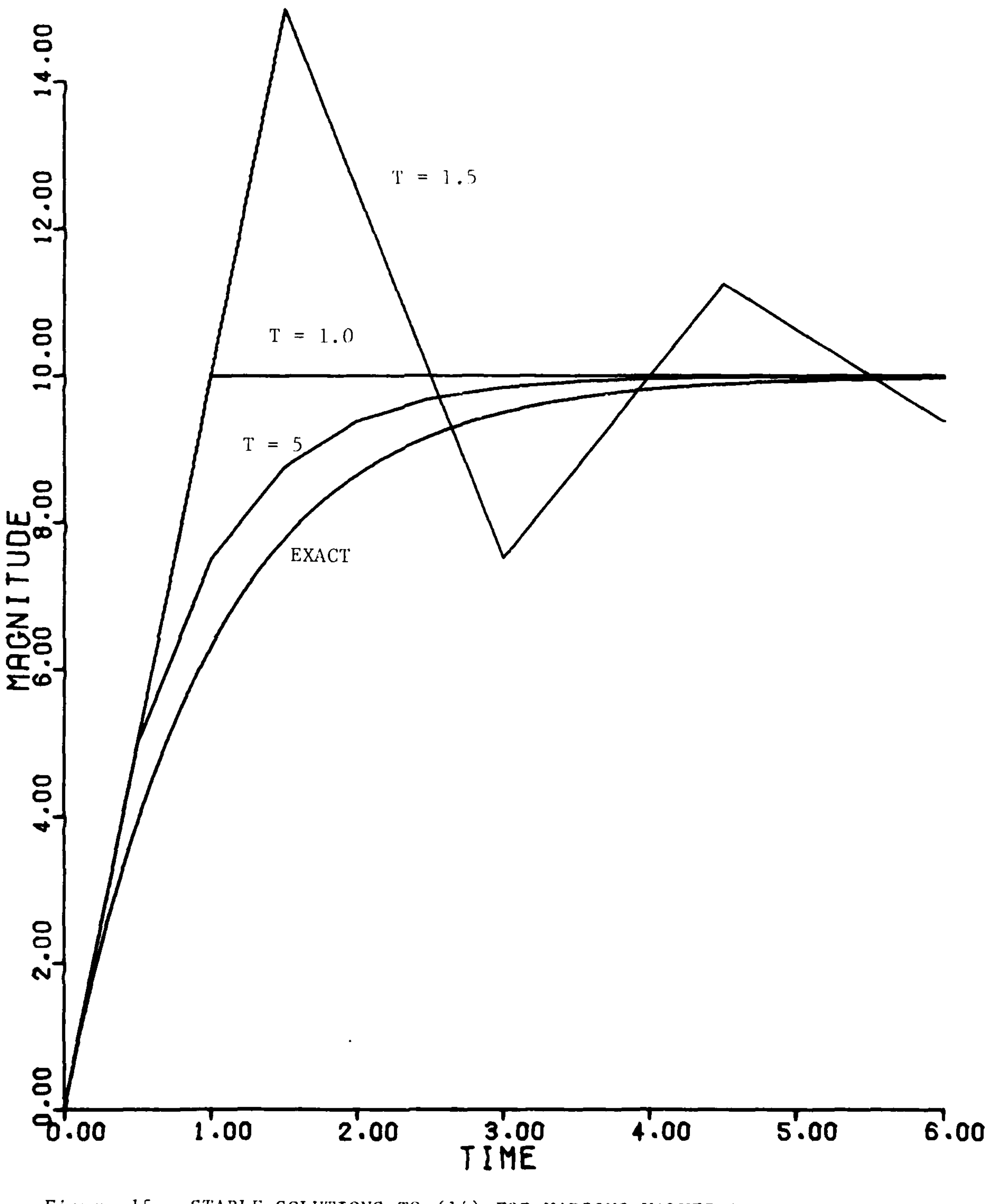


Figure 15. STABLE SOLUTIONS TO (14) FOR VARIOUS VALUES OF T.

V. CONCLUSIONS AND RECOMMENDATIONS

A digital structure has been designed, built and tested. The results show that a digital simulation machine can be an effective computing structure provided higher order numerical methods can be implemented. In general, the preliminary conclusions can be stated as:

1. The computing structure is simple and only requires five distinct modules.

2. The number of busses is proportional to the order of the differential equations.

3. For a general structure, the interconnection of the busses and device modules is the most complicated aspect of the structure.

4. The processor modules require only elementary software to configure them as a specific integrator.

5. The sequencing of the structure to carry out the solution of a set of state equations is simple. It has been done in software with only a few instructions, but can be done with hardware to improve the speed. The hardware solution is about the same complexity as one of the processor modules.

6. One typical system equation has been solved and evaluated. In general, higher order numerical methods give superior performance. For these methods 16-bit arithmetic is more accurate than the associated truncation errors.

7. Performance using less than state of the art digital technology will give real-time solutions of equations with frequency components up to 10 KHZ. The accuracy of the solutions will be approximately 1% of the maximum range. Improved technology could easily raise the frequency limit.

SPOCK I represents only the first step in a novel application of computer architecture to a special computing problem. The design was limited in scope and did not attempt to cover all classes of problems which could possibly be solved by such a structure. The following list presents a number of topics for further study and work to extend the field of knowledge in this area.

1. Software translator for automatic program set-up and sequencing.
2. Automatic scaling when variables exceed the overflow limit.
3. Extension to include time varying coefficients and nonlinear functions.
4. Extension to include a user interface for direct entry of an equation and a display of the solution.
5. Additional study on truncation errors, round-off errors and total error to ascertain the effect of additional poles or zeroes on these error components. The work thus far has concentrated on a dominant pair of complex poles in the differential equation. It is believed, though not verified, that this pair will set the limits on the maximum step size.
6. Additional study on performance improvements via alternate structures, custom hardware chips and different bussing concepts.

VI. REFERENCES

1. H. Brafman and B. J. Reuter, "An incremental computer", _IEEE Transactions on Computers_, Vol. C-26, No. 11, November 1977, pp. 1072-1081.
2. R.E.H. Bywater and W.P. Lovering, "A programmable extended resolution digital differential analyser", _The Radio and Electronic Engineer_, May 1972, pp. 203-212.
3. W. Forsythe, "A critique of a new digital differential analyser", _Simulation_, April 1975, pp. 97-102.
4. B.R. Gilbert and M.J. Morse, "Digital simulator replaces analog portion of hybrid computer", _Computer Design_, April 1976, pp. 91-97.
5. M.W. Goldman, "Design of a high speed DDA", _Proceedings 1965 Joint Computer Conference_, AFIPS, Vol. 27, Pt. I, pp. 929-949.
6. J. Hatvany, "The d.d.a. integrator as the interactive module of a variable structure process control computer", _Automatica_, Vol. 5, No. 1, January 1969, pp. 41-49.
7. G.P. Hyatt and S. Ohlberg, "Electrically alterable digital differential analyser", _A.F.I.P.S. Proceedings_, Spring Joint Computer Conference, 1968, pp. 161-169.
8. H. Kalis, H. Bantelmann, and A. Montag, "An all-digital completely software programmable hybrid computer", _Simulation of Systems_, L. Dekker (Ed.), North-Holland Publishing Company, 1976, pp. 277-286.
9. R.L.A. Kett and W.C. Rae, "A new development in operational computing systems", 6th International Hybrid Computation Meetings, pp. 793-797.
10. W.F. Lovering and R.E.H. Bywater, "An all digital hybrid computer", _1974 Conference of Computing Systems and Technology_, London, England, 28-31 October 1974, pp. 49-57.

11. R.B. McGhee and R.N. Neilsen, "The extended resolution digital differential analyser: A new computing structure for solving differential equations", IEEE Transactions on Computers, Vol. C-19, January 1970, pp. 1-9.

12. F. Cennaro and U. de Carlini, "A new ultra fast digital differential analyser employing multi-microprocessor's techniques", Microarchitecture of Computer Systems, R.W. Hartenstein and P. Zaks (Eds.), Euromicro, North-Holland Publishing Company, 1975, pp. 163-170.

13. J.L. Elshoff and P.T. Hulina, "The binary floating point digital differential analyser", Proceedings 1970 Fall Joint Computer Conference, AFIPS, pp. 369-376.

14. G.F. Graber and B.J. Fadden, "A next-generation hybrid computer system", Computer, July, 1976, pp. 55-62.

15. G. Bekey and W.J. Karplus, Hybrid Computation. New York, New York: John Wiley and Sons, Inc., 1968.

16. K. Ogata, State Space Analysis of Control Systems, Englewood Cliffs, NJ; Prentice-Hall, 1967, pp. 106-171.

17. Intel Corporation, Intel Series 3000 Reference Manual. Santa Clara, Calif; Intel Literature Department, 1976.

18. B. Carnahan, H.A. Luther, J.O. Wilkes, Applied Numerical Analysis. New York, NY: McGraw-Hill, 1969, pp. 346-347, 363-366, 386-390.

19. J.F. Richardson and J.A. Gaunt, "The deferred approach to the limit", Trans. Royal Society, London, England, Vol. 226A, 1927, p. 300.

20. M.L. James, G.M. Smith and J.C. Wolford, Applied Numerical Methods for Digital Computation with Fortran. Seraton, PA: International, 1967, pp. 386-388.

21. G.A. Bekey and W.J. Karplus, Hybrid Computation, New York, New York: John Wiley & Sons, Inc., 1968, pp. 99-106.

22. A. Durling, Computational Techniques: Analog, Digital and Hybrid Systems, New York, New York: Intext Educational Publishers, 1974, pp. 186-189.

23. B. Carnahan, H.A. Luther, J.O. Wilkes, Applied Numerical Analysis. New York, New York: McGraw-Hill, 1969, pp. 363-365.

24. A. Durling, Computational Techniques: Analog, Digital and Hybrid Systems. New York, New York: Intext Educational Publishers, 1974, pp. 24-27.